



**ISEL**

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA



# Instituto Superior de Engenharia de Lisboa

DEPARTAMENTO DE  
Engenharia da Electrónica das  
Telecomunicações e dos Computadores

# Sistemas Computacionais Distribuídos

WEB - Geração de Páginas dinamicamente - Servlet's



# Índice

 Ambiente executivo das Servlet's

 Construção de Servlet's

 Exemplo Simples

 Ciclo de Vida das Servlet

 Estado entre Pedidos

 Exemplos



# Enquadramento

- Geração de conteúdos dinâmicos no servidor:
  - Common Gateway Interface – CGI
  - Extensões ao servidor – ISAPI e NSAPI
  - Java Servlet's
  - Server Side Includes (SSI)
  - Server Side JavaScript (SSJS)
  - Active Server Pages (ASP)
  - JavaServer Pages (JSP)



# Introdução

- Uma Servlet permite estender a funcionalidade de um servidor web do mesmo modo que um CGI.
  - Não é mais que um objecto Java cujos métodos são invocados pelo servidor para tratar pedidos HTTP.
- 
- **Applet's Vs Servlet's**
    - Applet – Classe java que é executada do lado do cliente
    - Servlet – Classe java que é executada do lado do servidor



# Introdução (cont.)

- As Servlet's são a forma de em Java criar aplicações Web enabled;
- Cada Servlet é como se fosse um mini servidor Web aumentando as suas capacidades fornecendo funcionalidades adicionais;
- Estas novas funcionalidades podem servir para criar:
  - Websites de E-commerce
  - Front-end para Bases de dados
  - Conversor de imagens
  - Etc...
- Uma Servlet recebe um pedido HTTP e retorna uma resposta HTTP

À semelhança de um  
CGI



# Porquê Servlets? (Vantagens)

## ■ As Servlets são:

- **Persistentes** – Ao contrário dos CGI's o ciclo de vida das servlets estende-se para além de um pedido HTTP.
- **Simples** – São simples de escrever.
  - Quem realiza as tarefas mais complicadas é o servlet *container*.
  - Java é uma linguagem de alto nível que impede a manipulação directa da memória do sistema, reduzindo a probabilidade de bugs graves.
- **Flexíveis** – Tem acesso a todas as API's Java. Estas são bem definidas e com bastante experiência e suporte pela comunidade Java.
- **Eficientes** – Por cada pedido HTTP são utilizadas tarefas Java e não novos processos.
- **Portáveis** – Executam-se na **Máquina Virtual Java (JVM)**.
- **Seguras** – Executam-se no lado do servidor daí serem seguras para os clientes.



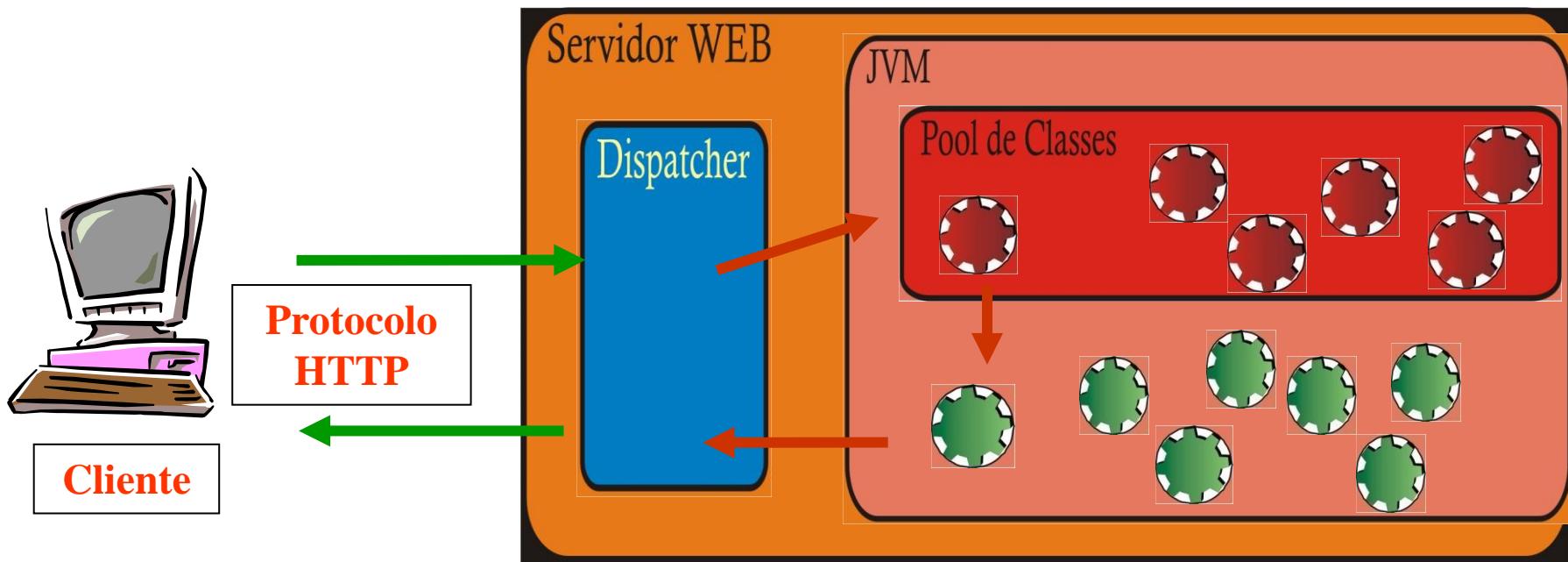
# Introdução / Enquadramento

- As **Servlet's** e as **JavaServer Pages** (JSP) são um subconjunto da coleção de API's do lado do servidor, de nome **Java Enterprise Edition** (JEE);
- As tecnologias **JEE** foram desenvolvidas para providenciar uma infra-estrutura escalável e fácil de manter;
- As **Servlet's** e as **JavaServer Pages** juntas formam a camada de apresentação das *web applications* JEE, enquanto que os **Enterprise JavaBeans** (EJB) são utilizados para desenvolver as camadas de lógica de dados;  


Abordagem que vamos estudar
- Para aplicações de complexidade simples/moderada as **Servlet's** e as **JSP's** podem só por si, ou com a ajuda de **JavaBeans normais**, servir todas as necessidades.



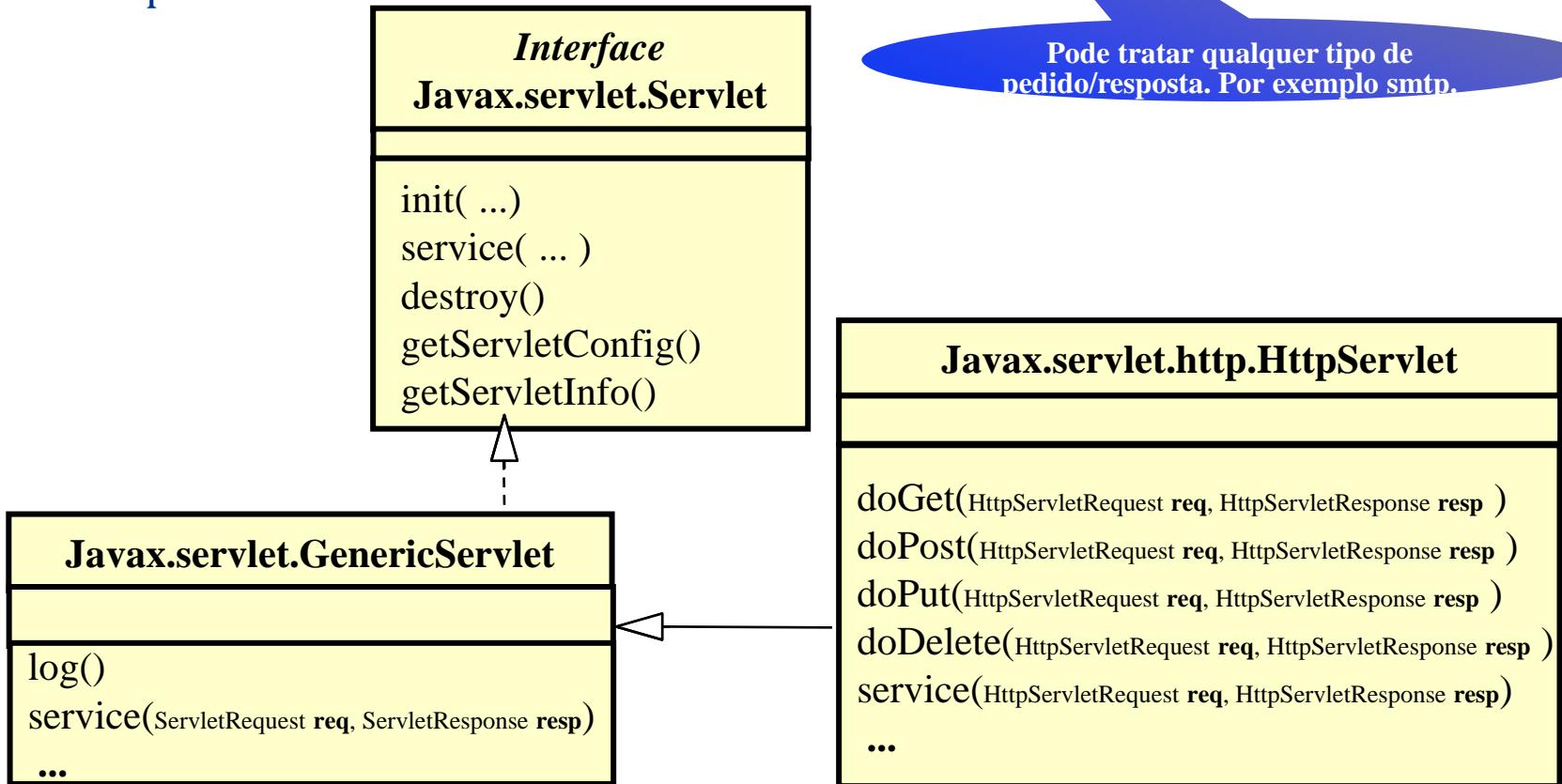
# Ambiente executivo das Servlet's





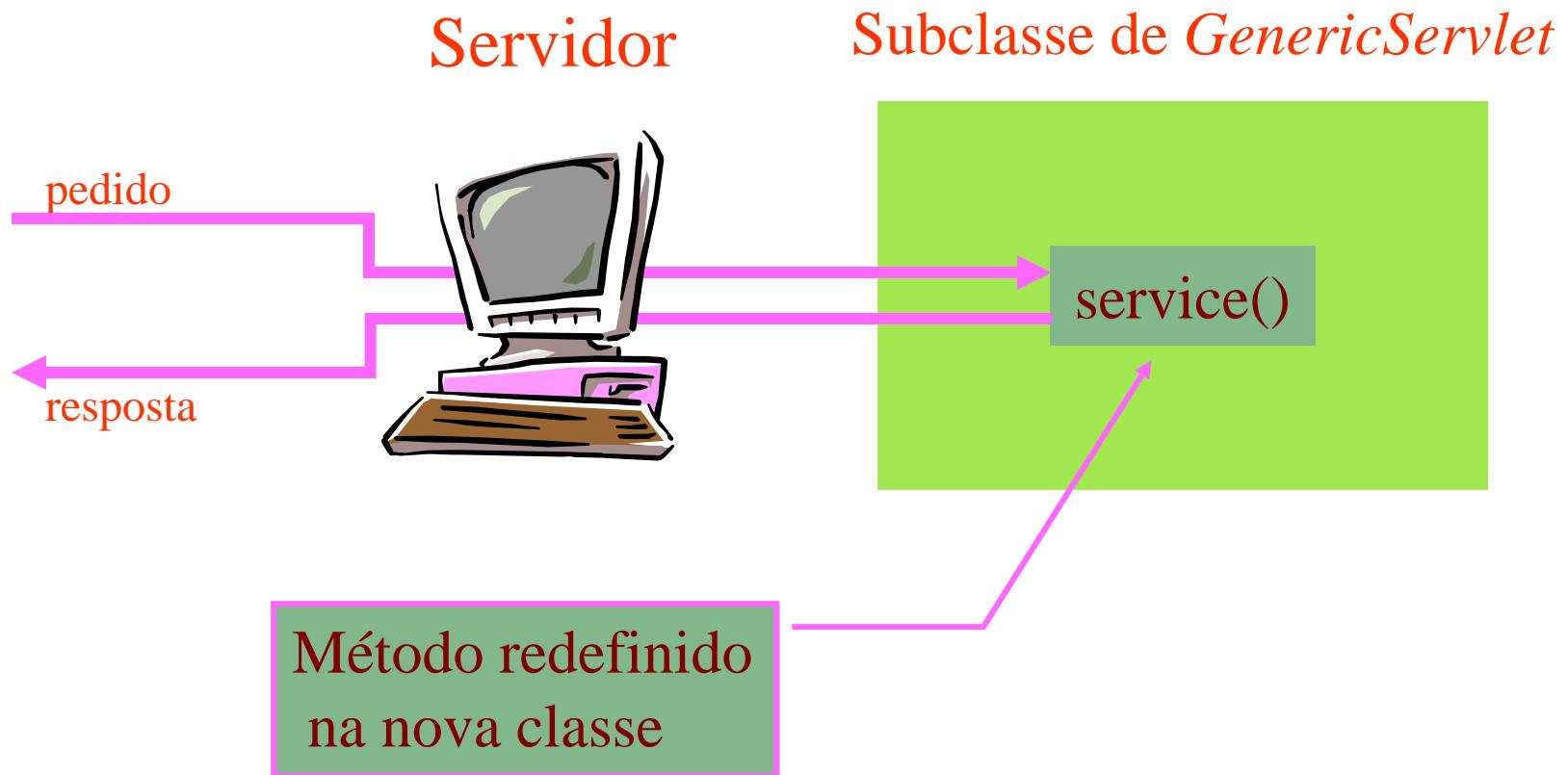
# Construção de Servlet's

- Uma Servlet é apenas uma classe Java que implementa a interface `javax.servlet.Servlet`.
- Particularizando para o protocolo HTTP temos que uma Servlet será uma subclasse da classe `javax.servlet.http.HttpServlet`.
- Uma Servlet não tem método `main()`, apenas disponibiliza alguns métodos que são invocados pelo servidor Web.



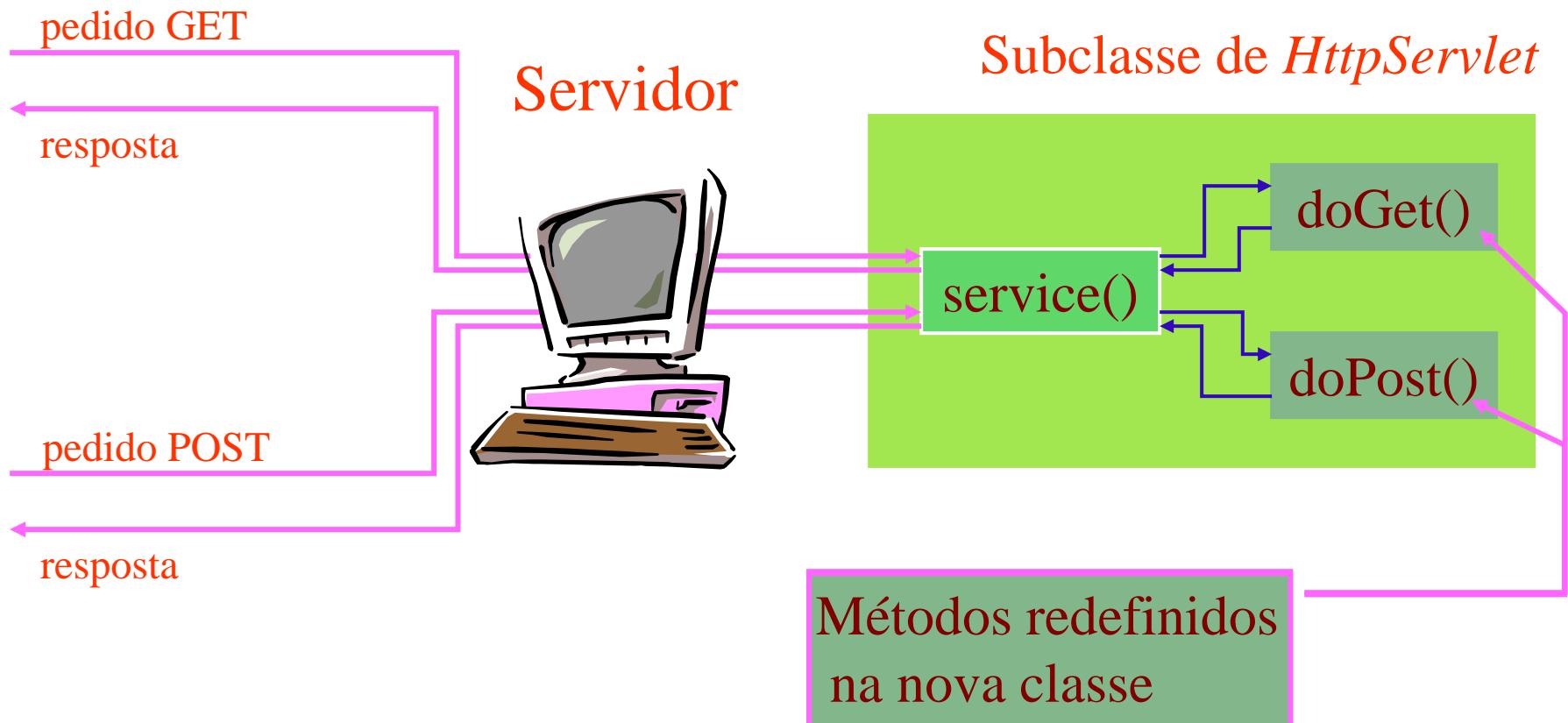


# Construção de Servlet's





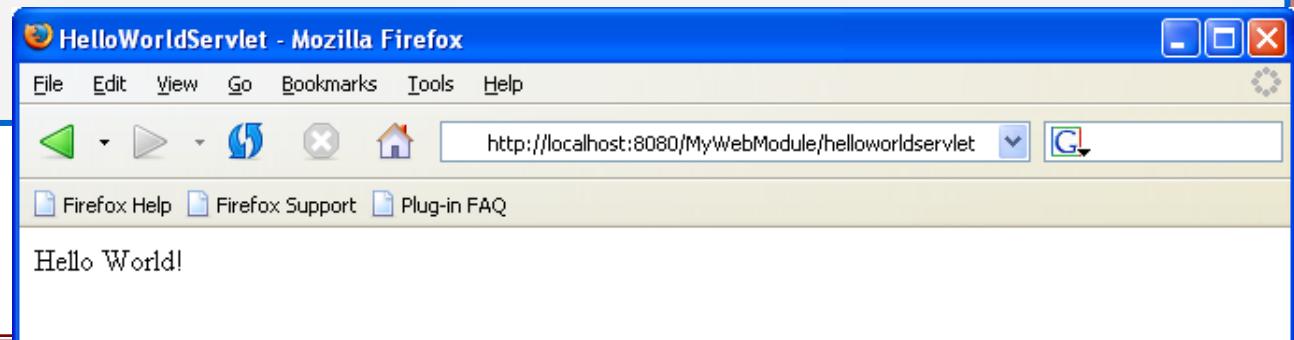
# Construção de Servlet's





# Exemplo – Hello World!!!

```
public class HelloWorldServlet extends javax.servlet.http.HttpServlet {  
  
    //Process the HTTP Get request  
  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
            throws ServletException, IOException {  
  
        response.setContentType("text/html");  
  
        PrintWriter out = response.getWriter();  
  
        out.println("<html>");  
  
        out.println("<head><title>HelloWorldServlet</title></head>");  
  
        out.println("<body>");  
  
        out.println("<p>Hello World! </p>");  
  
        out.println("</body></html>");  
    }  
}
```





# Exemplo 2 - Hello To!!!

```
<html>
<head>
<title>Formulário para o nome</title>
</head>
<body>
<hr/>
<font color="#000080"><h1>Formulário com o método http GET</h1></font>
<form method="GET" action="hellotoservlet">
    <h1>Insira o seu nome:</h1><input type="text" name="nome"/>
    <input type="submit" name="submit" value="Submeter"/>
    <input type="reset" name="reset" value="Limpar"/>
</form>

<hr/>
<font color="#000080"><h1>Formulário com o método http POST</h1>
<form method="POST" action="hellotoservlet">
    <h1>Insira o seu nome:</h1><input type="text" name="nome"/>
    <input type="submit" name="submit" value="Submeter"/>
    <input type="reset" name="reset" value="Limpar"/>
</form>

</body>
</html>
```

Página *html* (*index.html*) com o formulário que recolhe os dados para serem processados pela *servlet*

The screenshot shows a Mozilla Firefox window with two identical forms displayed side-by-side. Both forms have the title "Formulário para o nome". The top form is labeled "Formulário com o método http GET" and the bottom form is labeled "Formulário com o método http POST". Both forms contain the text "Insira o seu nome:" followed by a text input field, a "Submeter" button, and a "Limpar" button. The address bar shows the URL <http://localhost:8080/MyWebModule/index.html>.



# Exemplo 2 - Hello To!!!

```
public class HelloToServlet extends HttpServlet {  
    //Process the HTTP Get request  
  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        String rcvNome = request.getParameter("nome");  
        if (rcvNome == null) {  
            rcvNome = "John Doe"; // default value  
        }  
  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println("<html>");  
        out.println("<head><title>HelloToServlet</title></head>");  
        out.println("<body>");  
        out.println("<h1>Hello " + rcvNome + "</h1>");  
        out.println("<p><i><b>[" + new java.util.Date() + "]</b></i> A Servlet recebeu um pedido http: " +  
                   request.getMethod() + ".</p>");  
        out.println("</body></html>");  
    }  
  
    //Process the HTTP Post request  
  
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,  
    IOException {  
        doGet(request, response);  
    }  
}
```



# Exemplo 2 – Hello To!!!

**Hello Diogo & Cajó**

*[Sun Nov 21 17:06:05 GMT 2004]* A Servlet recebeu um pedido http: GET.

Foi submetido o primeiro formulário.  
Enviou-se para o servidor um pedido http  
**GET** com os dados concatenados no URL

**Foi submetido o segundo formulário.  
Enviou-se para o servidor um pedido http  
**POST** com os dados no corpo de dados da  
mensagem HTTP.**

Código fonte do HTML gerado

**Source of: http://localhost:8080/MyWebModule/hellotoservlet?nome=Diogo+%26+Caj%F3 - Mozilla Firefox**

```
<html>
<head><title>HelloToServlet</title></head>
<body>
<h1>Hello Diogo & Cajó</h1>
<p><i><b>[Sun Nov 21 17:06:05 GMT 2004]</b></i> A Servlet recebeu um pedido http: GET.</p>
</body></html>
```



# Ciclo de Vida das Servlet's

- É instanciada num contentor próprio:
  - É carregada para memória pela primeira vez para servir o 1º pedido que a referencia; permanece instanciada para os pedidos que, eventualmente, se sigam; pode ser removida se não for usada a partir de um dado tempo sem ter pedidos. Os pedidos são executados através de tarefas (existe concorrência).
  - Partilham entre si a mesma máquina virtual. No entanto a linguagem Java impede que uma Servlet aceda aos dados doutra;
- Os servidores Web são livres de implementar o motor de suporte às Servlets desde que:
  - Criem e iniciem as Servlets;
  - As Servlets tratem zero ou mais pedidos feitos por clientes;
  - Destruam a Servlet e efectuem o *garbage collection*.
- A máquina virtual Java pode residir:
  - No próprio servidor Web (se o servidor for implementado em Java) (e.g. Tomcat *standalone*);
  - Numa máquina virtual Java a correr num processo, independente, do servidor Web  
(e.g. Apache e Tomcat interligados com um conector).

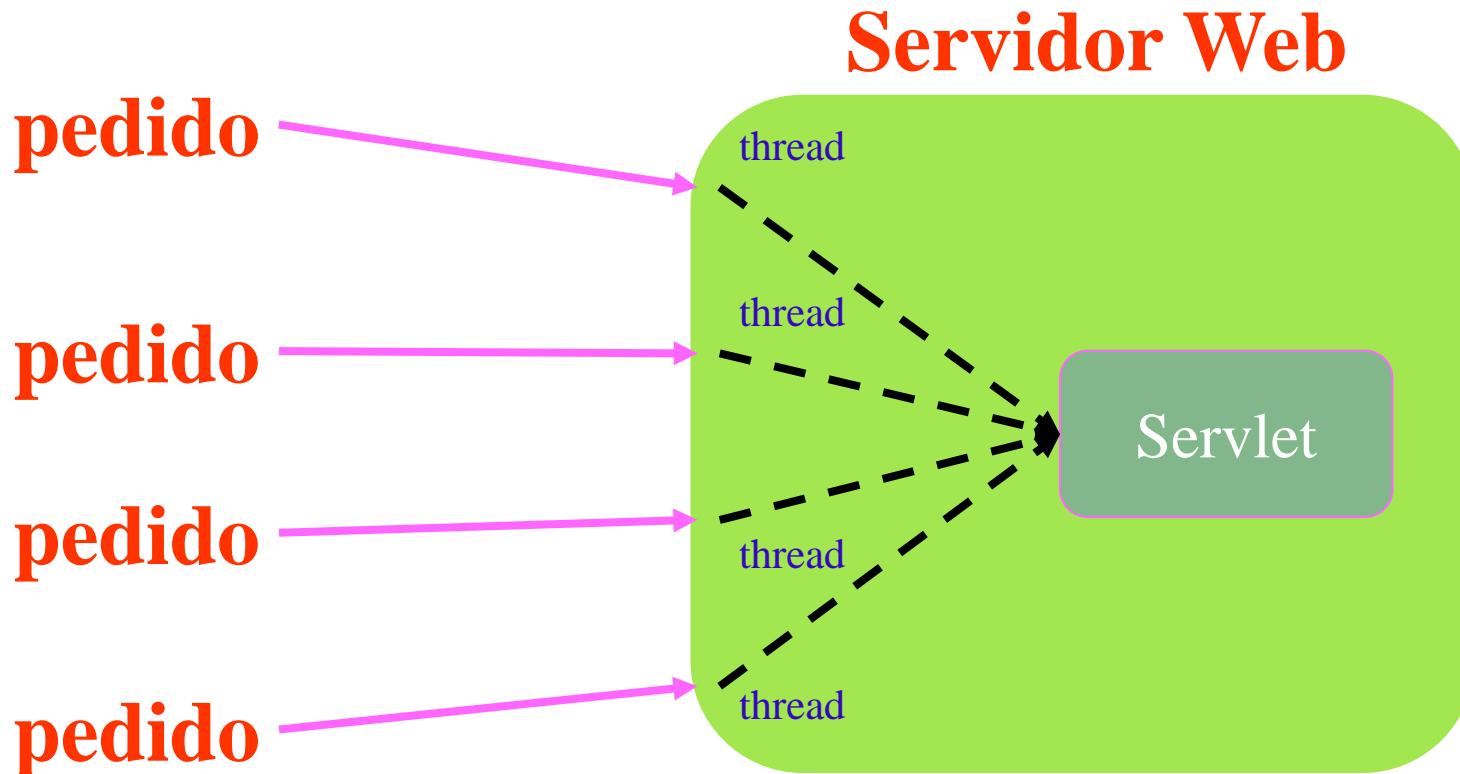


# Ciclo de Vida das Servlet's

- As Servlets não têm “construtor”! Ou melhor têm mas não se utiliza para iniciar um objecto em particular.
- Em alternativa a API oferece dois métodos:
  - `init()` – chamado quando é criada uma nova instância da Servlet (instanciada no contentor);
  - `destroy()` – quando a Servlet é destruída (removida do contentor);
- Estes métodos são invocados pelo servlet *container* (contentor das servlets).
- Para cada pedido http recebido é atribuída uma nova tarefa e esta invoca o método:  
**`service (HttpServletRequest, HttpServletResponse) ;`**
  - `HttpServletRequest`
    - © Objecto que representa e guarda toda a informação referente ao pedido HTTP efectuado pelo browser.
  - `HttpServletResponse`
    - © Objecto que representa e guarda toda a informação referente à resposta HTTP a ser devolvida para o *browser*.



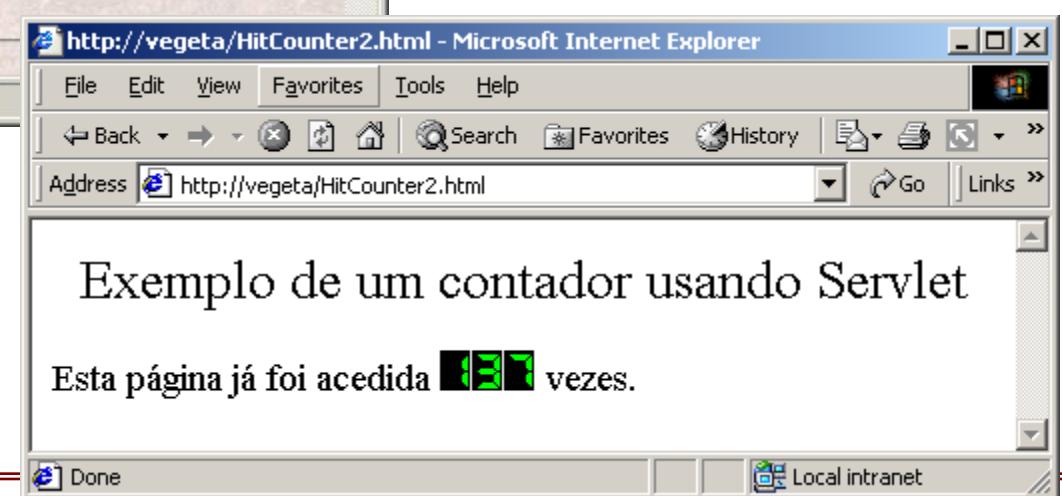
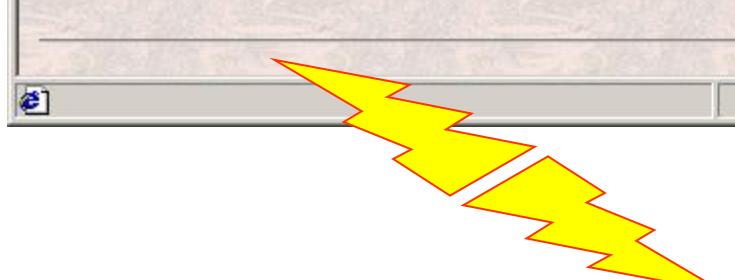
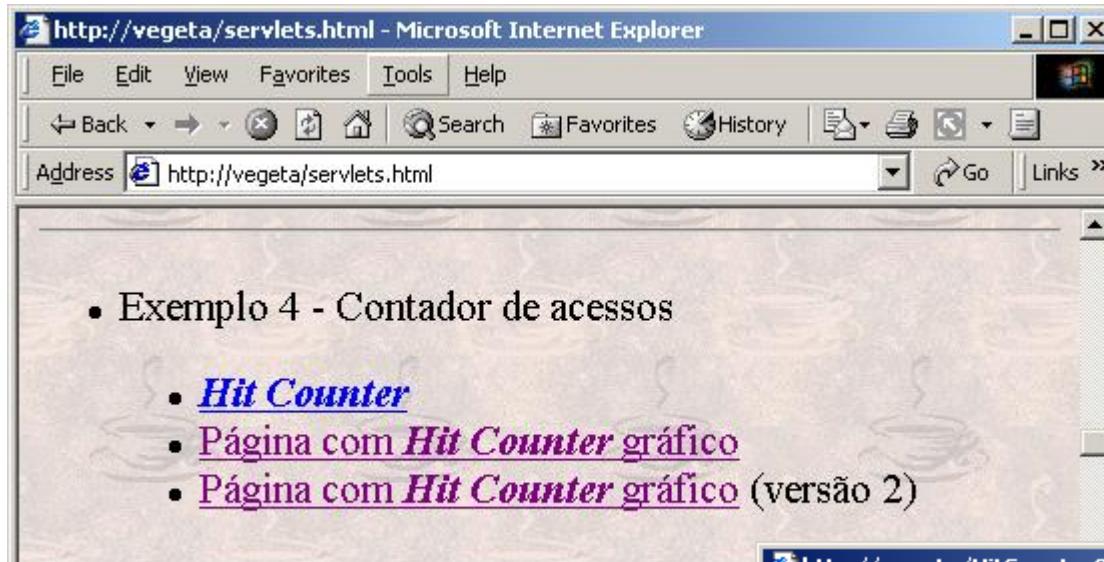
# Ciclo de Vida das Servlet's



- Uma Servlet é instanciada no 1º acesso, ou no arranque do servidor, e essa mesma instância será utilizada por todas as threads lançadas para tratar os vários pedidos que forem recebidos pelo servidor web.



# Ciclo de Vida das Servlet's





# Ciclo de Vida das Servlet's

```
import java.io.*;
import java.awt.*;
import java.lang.*;
import javax.servlet.*;
import javax.servlet.http.*;
import Acme.JPM.Encoders.*;

public class HitCounterGh2 extends HttpServlet {
    static String DIR_IMAGE = "/images/odometer/";
    int count = 0;

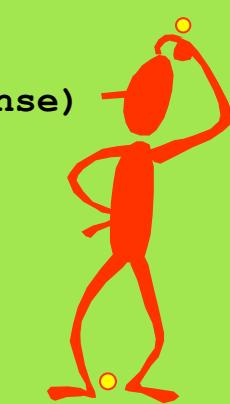
    synchronized private String getCount() {
        return "" + (++count);
    }
}
```



# Ciclo de Vida das Servlet's

Que acontece se o servidor é desligado ?

```
private Image getImageCount(String stringCount) throws ServletException {  
    //...  
}  
  
public void doGet(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException {  
  
    response.setContentType("image/gif");  
  
    GifEncoder encoder = new GifEncoder( getCount() ,  
                                         response.getOutputStream());  
    encoder.encode();  
}  
};
```



Vou utilizar os métodos **init** e **destroy** para ler e guardar a contagem



# Ciclo de Vida das Servlet's

```
public void init(ServletConfig config) throws ServletException {
    super.init(config);

    try {
        FileReader fileReader = new FileReader("HitCounterGh2.dat");
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        count = Integer.parseInt( bufferedReader.readLine() );
        fileReader.close();
    }
    catch (Exception e) {
        count = 0;
    }
}
```



# Ciclo de Vida das Servlet's

```
public void destroy() {  
    saveCount();  
}  
  
private void saveCount() {  
    try {  
        FileWriter fileWriter = new FileWriter("HitCounterGh2.dat");  
        String stringCount = Integer.toString(count);  
        fileWriter.write(stringCount, 0, stringCount.length());  
        fileWriter.close();  
    }  
    catch (Exception e) {}  
}
```



# Web Application

- O conceito de ***web Application*** foi introduzido na especificação das **Servlet's 2.2**.
- De acordo com esta especificação, uma ***web Application*** é uma colecção de **servlet's, páginas html, classes** e qualquer outro **recurso** que possa ser empacotado para correr em vários ***web containers*** de diversos fabricantes.
- Uma definição mais prática é dizer que uma ***web Application*** é tudo o que reside na camada **WEB** de uma aplicação.
- Uma ***aplicação web*** pode ser constituída pelos seguintes itens:
  - Servlet's
  - JavaServer Pages (JSP)
  - Classes utilitárias
  - Documentos estáticos que incluem html, imagens, etc...
  - Bibliotecas (ficheiros .jar)
  - Meta informação que descreve a aplicação *web*



# Web Application (cont.)

- Na prática uma **Web Application** corresponde a uma directória com o seu nome e com a respectiva estrutura de directorias que armazenam os seus componentes

<b>Directória:</b>	<b>Contém:</b>
/WebAppName/	É a directória raiz da aplicação web ( <i>Web application</i> ). Todas as páginas HTML e JSP são armazenadas aqui.
/WebAppName/WEB-INF/	É aqui que está localizado o descritor de instalação da aplicação web (web.xml). Note que esta directória não pertence à parte pública da aplicação. Nenhum ficheiro nesta directória pode ser acedido directamente por parte do cliente.
/WebAppName/WEB-INF/classes/	Nesta directória são colocadas as servlet's (.class) e todas as classes utilitárias.
/WebAppName/WEB-INF/lib/	Esta directória deve conter todos os Java Archive Files (.JAR) que a aplicação Web necessita. Por exemplo, onde se deve colocar o .jar que contém o driver JDBC para aceder a bases de dados.



# O deployment descriptor da web application

■ O *deployment descriptor* é um ficheiro em **XML** (*Extensible Markup Language*) com o seguinte nome:

■ `/WebAppName/WEB-INF/web.xml`

■ A informação que este contém inclui os seguintes elementos:

- Os parâmetros iniciais (init) do ServletContext
- Configuração das sessões (session)
- Definição das SERVLET's e JSP's
- Mapeamento das SERVLET's e JSP's
- Mapeamento de Mime types (mapeamento [extensão do fich. <=> mime type])
- Lista das páginas iniciais (ex: index.html)
- Páginas de erros
- Conteúdo localizado

```
<web-app>
    <display-name>WebAppName</display-name>
    <description>Aplicacao exemplo</description>
    <session-timeout>30</session-timeout>
    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>HelloWorldServlet</servlet-class>
        <load-on-startup>0</load-on-startup>
    </servlet>
    ...
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

**/WebAppName/WEB-INF/web.xml**

Classe que implementa a servlet  
HelloWorldServlet.class

Padrão de letras que identificam  
no URL a servlet hello



# Web ARchive - WAR

## ■ Empacotamento de uma **Web Application** :

- Agora que já sabemos o que é uma **web application** podemos empacota-la para efectuar o seu *deployment* (instalação) num **web container**.
- O método standard de empacotar uma **web application** é utilizando um **Web ARchive file** (WAR).
- Podemos criar um ficheiro WAR utilizando a ferramenta "**Java archiving tool**" jar.

```
$jar cvf myTestWebApp.war .  
added manifest  
adding: WEB-INF/(in = 0) (out= 0) (stored 0%)  
adding: WEB-INF/classes/(in = 0) (out= 0) (stored 0%)  
adding: WEB-INF/classes/helloworldservlet/(in = 0) (out= 0) (stored 0%)  
adding: WEB-INF/classes/helloworldservlet/HelloWorldServlet.class(in = 1290) (out= 635) (deflated 50%)  
adding: WEB-INF/web.xml(in = 530) (out= 266) (deflated 49%)
```

Na directória raiz da Web Application

Como resultado é criado um ficheiro myTestWebApp.war, que contém toda a web application comprimida.



# Apache Jakarta Tomcat

- O Tomcat é o servlet container utilizado na implementação oficial de referência das tecnologias Java Servlet and JavaServer Pages

Servlet/JSP Spec	Tomcat version	
2.5/2.1	6.0.16	
2.4/2.0	5.5.26	
2.3/1.2	4.1.37	
2.2/1.1	3.3.2	

**Servlets:** <http://java.sun.com/products/servlet/>

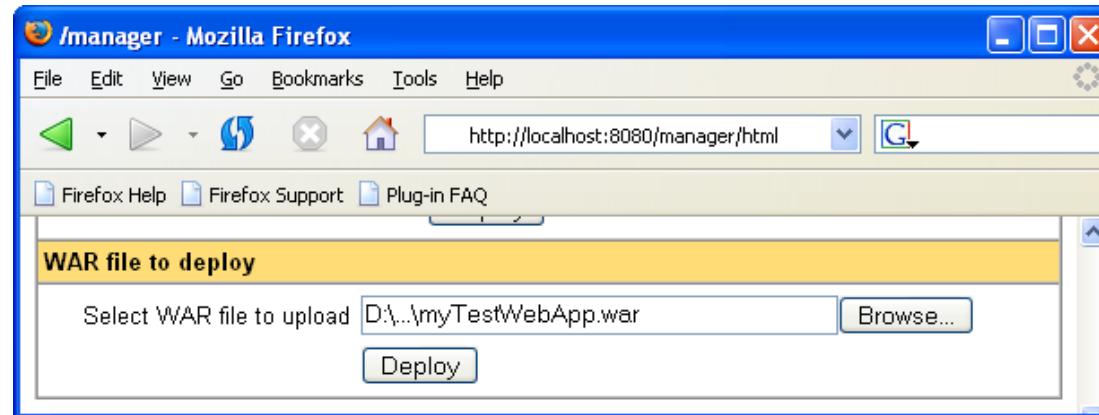
**Especificação:** <http://java.sun.com/products/servlet/reference/api/index.html>

**Tomcat:** <http://tomcat.apache.org/>



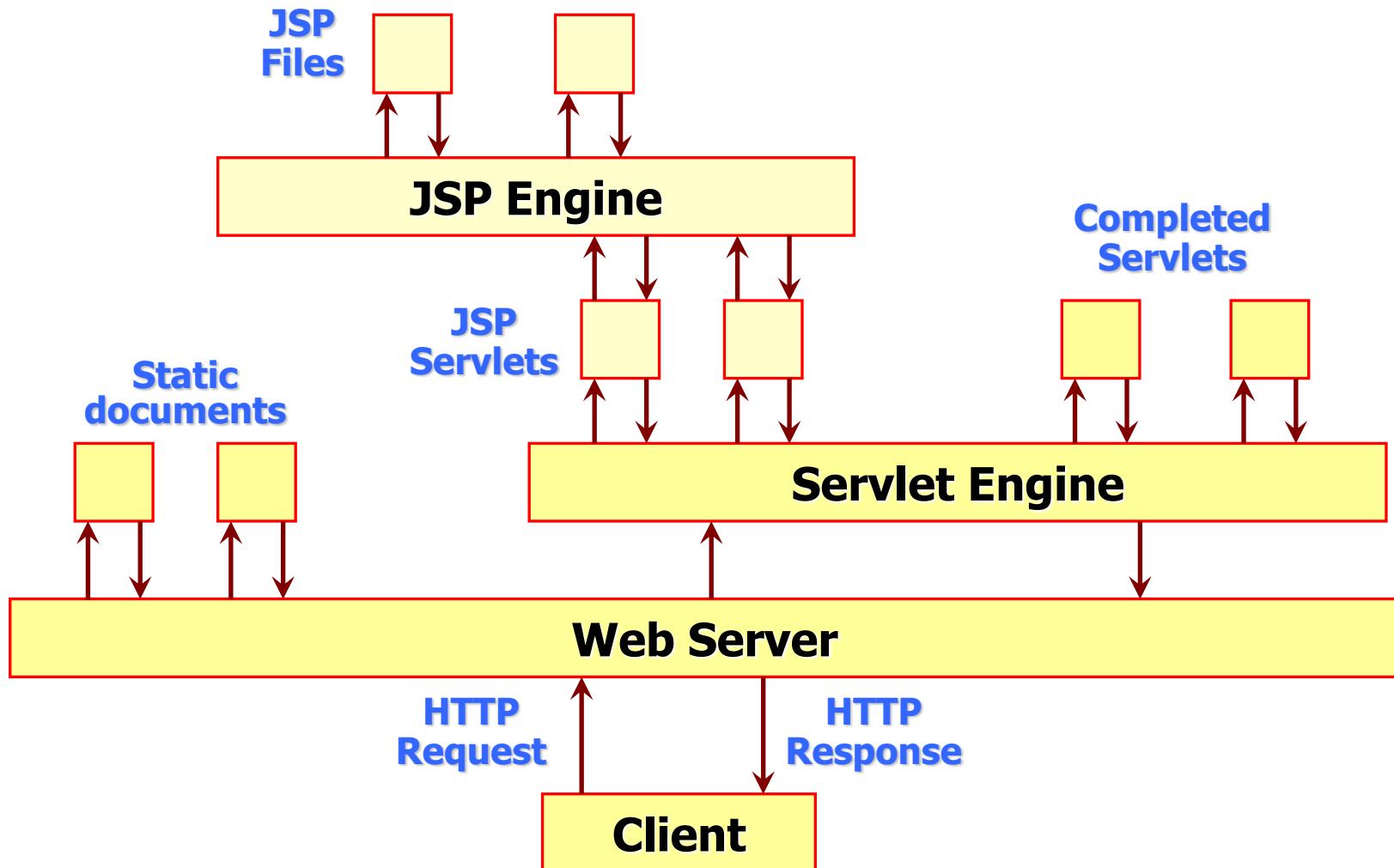
# Efectuar o *Deployment* de uma *Web App.*

- Existem várias formas de efectuar o *deploy* de uma *web Application* e varia consoante o *Servlet container* que está a ser utilizado:
  - Copiar a directória da web application para a directória (necessita de um restart ao tomcat):
    - **%Tomcat\_Home%/webapps/**
- ★ Utilizar o Web ARchive (war)
- ★ Via interface *web* do *manager* do *tomcat*



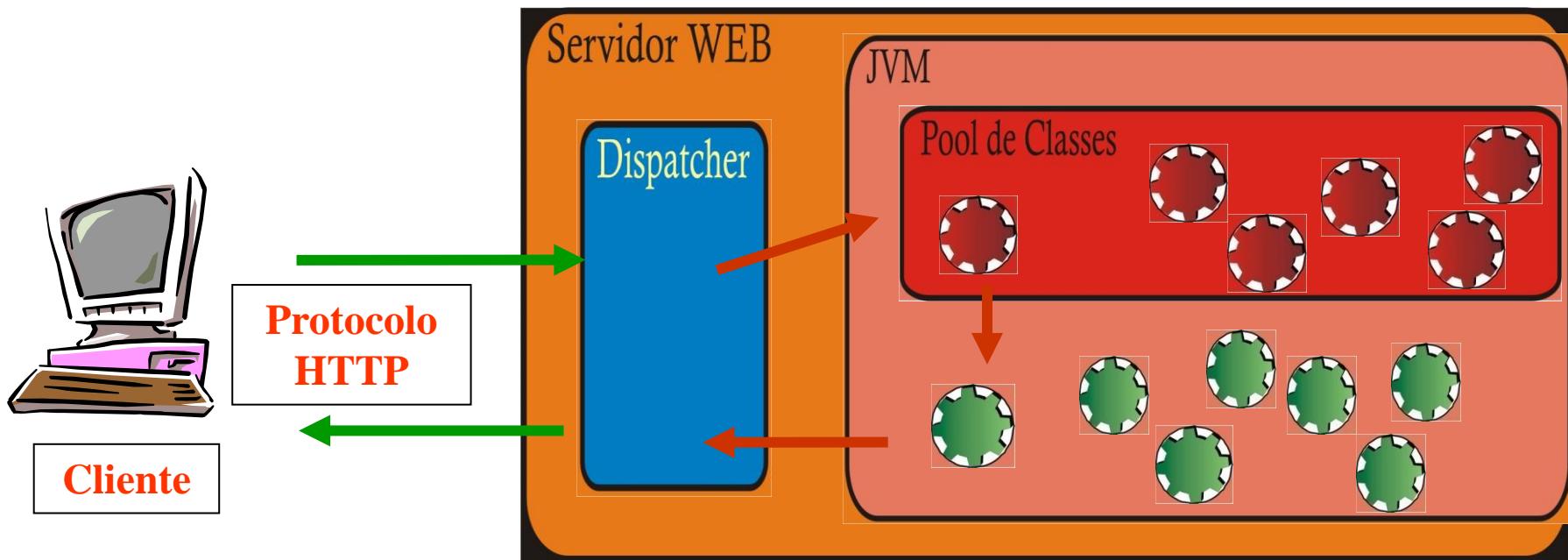


# Arquitectura interna de um Servlet Container





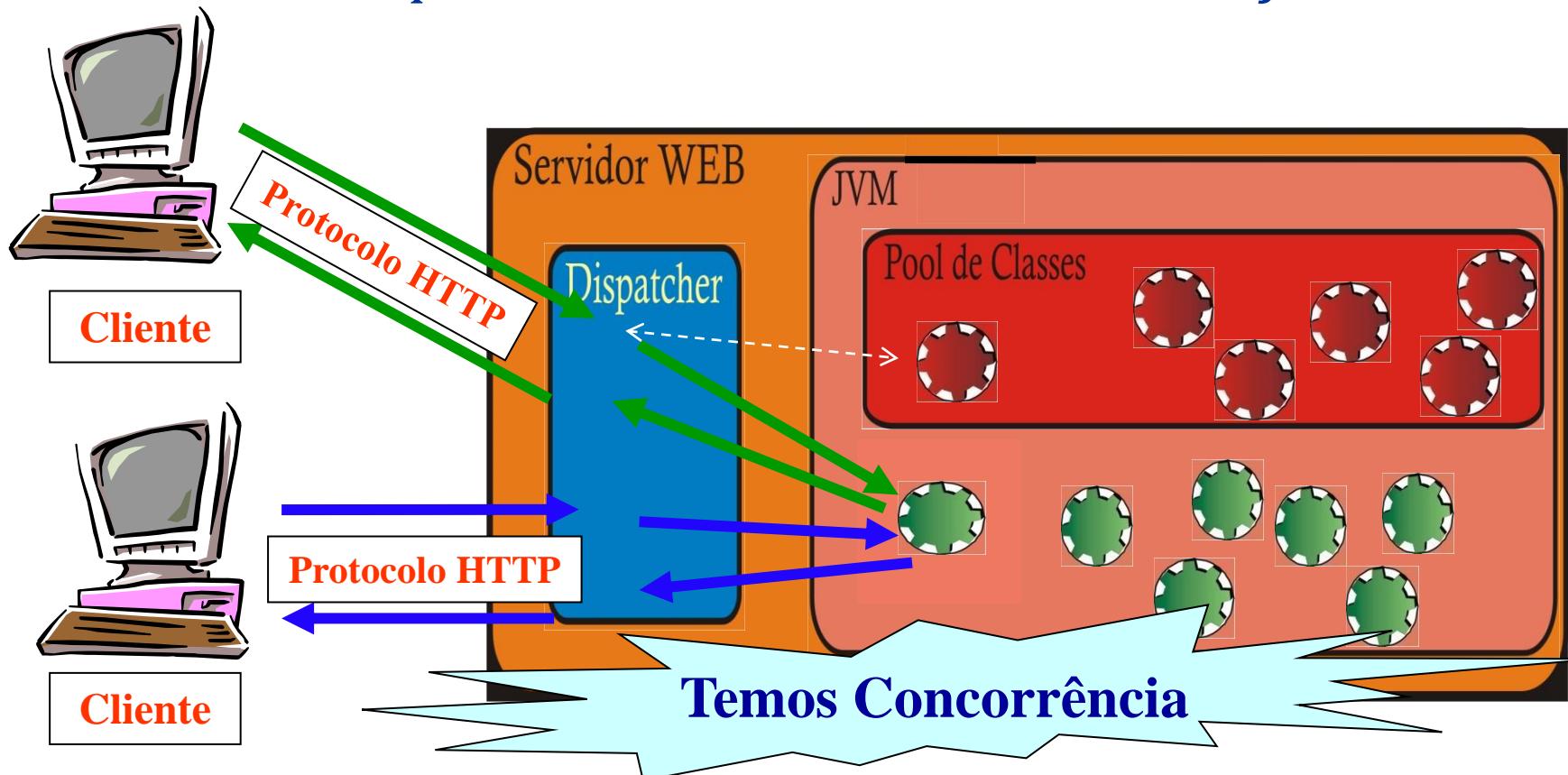
# Ambiente executivo das Servlet's





# Ambiente Multi-Thread

- Por cada pedido **HTTP** que o **Tomcat** recebe cria uma nova **tarefa** e invoca o método **service()** da Servlet correspondente. Existe apenas uma instância da Servlet. O tomcat assume que todas as Servlets são ***thread safe***.





# Ambiente Multi-Thread

■ Para resolver o problema da concorrência temos as seguintes hipóteses:

■ Tratar explicitamente a concorrência utilizando os mecanismos de sincronização que temos à disposição.

O Tomcat por omissão assume esta hipótese.

■ Criar uma Servlet que implemente o interface ***SingleThreadModel***. É apenas uma marker para informar o Servlet container que a Servlet não é ***ThreadSafe*** e que este terá de garantir que não existem duas tarefas a invocar, simultaneamente, o método `service` da mesma instância. Existem duas hipóteses possíveis:

- © Ou mantêm uma instância que atende os pedidos à vez;
- © Ou cria uma pool de instâncias para cada uma atender uma tarefa.

O Tomcat cria uma pool de instâncias para tratar a concorrência. Atenção no uso de variáveis da instância em particular.



# Troca de Informação browser/servidor/servlet

- Além da informação que o utilizador “gerou” as Servlets têm à sua disposição informação sobre:
  - O servidor (web);
  - O cliente (browser);
  - O pedido;
- Os métodos da `classe javax.servlet.http.HttpServlet` e suas ascendentes permitem obter essas informações.



# Troca de Informação browser/servidor/servlet

## ■ Informação sobre o servidor Web:

- Nome do servidor;
- Porto do servidor;
- Versão do servidor;
- Directória raiz do servidor;



# Troca de Informação browser/servidor/servlet

## ■ Informação sobre o cliente (*browser*):

- Nome do *host* cliente;
- Endereço do cliente;
- Tipo de autenticação;
- Nome do utilizador;



# Troca de Informação browser/servidor/servlet

## ■ Informação sobre o pedido:

- Protocolo usado (versão HTTP);
- Método usado;
- Tipo do conteúdo;
- Dimensão do conteúdo;
- Tipos de MIMEs aceites pelo *browser*;
- Informação sobre software do *browser*;
- Link usado pelo cliente;



# Exemplo - Snooping HTTP Headers

```
public class SnoopingHttpHeadersServlet extends HttpServlet {  
    //Process the HTTP Get request  
  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        response.setContentType("text/html");  
  
        PrintWriter out = response.getWriter();  
  
        out.println("<html>");  
        out.println("<head><title>SnoopingHttpHeadersServlet</title></head>");  
        out.println("<body>");  
        out.println("<hr><H1> Os cabe&ccedil;ahos do pedido HTTP: </H1><hr>");  
        Enumeration enumHeader = request.getHeaderNames();  
        while (enumHeader.hasMoreElements()) {  
            String name = (String) enumHeader.nextElement();  
            String value = request.getHeader(name);  
            out.println("<p><font color=\\"#000080\\">" + name +  
                        "</font>: <font color=\\"#008000\\">" + value + "</font></p>");  
        }  
        out.println("<hr/>");  
        out.println("</body></html>");  
    }  
}
```



# Exemplo – Snooping HTTP Headers

SnoopingHttpHeadersServlet - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8083/MyWebModule/snoopinghttpheadersservlet

Firefox Help Firefox Support Plug-in FAQ

## Os cabeçalhos do pedido HTTP:

```
host: localhost:8083
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7) Gecko/20040614 Firefox/0.9
accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
accept-language: en-us,en;q=0.5
accept-encoding: gzip,deflate
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
keep-alive: 300
connection: keep-alive
```

Done



# Estado entre Pedidos

- O protocolo HTTP é um protocolo sem estado entre pedidos;
- Assim sendo como é que se consegue guardar informação de estado?

**Nota: Não se pode usar o endereço IP pois o endereço pode ser o de um *proxy* que serve múltiplos clientes.**





# Estado entre Pedidos

- Uma das maiores vantagens de utilizar as Servlets é a facilidade de transformar, transparentemente, o protocolo HTTP (*stateless*), numa sequência de actividades possibilitando uma *web application* parecer uma aplicação normal.
- Utilizando Servlets existem vários métodos para manter o estado entre pedidos:
  - Autenticação de utilizadores;
  - Campos escondidos;
  - Rescrita dos URLs;
  - Utilização de *cookies*;
  - Usando a API (*Session Tracking*).



# Estado entre Pedidos

## ■ Autenticação de utilizadores:

- O administrador do servidor Web restringe uma área do servidor;
- No caso de o utilizador ser válido, pode-se obter o nome do utilizador através do método **getRemoteUser()**;

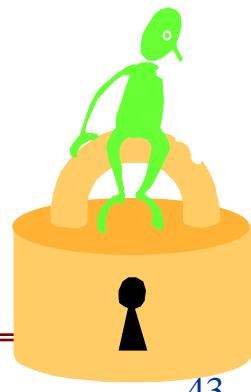




# Estado entre Pedidos

## ■ Autenticação de utilizadores:

- É simples de implementar;
- Funciona de um modo independente do local onde o cliente se encontra;
- É necessária uma conta para cada utilizador que acede ao recurso.





# Estado entre Pedidos

```
public class AutenticacaoDeUtilizadoresServlet extends HttpServlet {  
    //Process the HTTP Get request  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println("<html>");  
        out.println("<head><title>AutenticacaoDeUtilizadoresServlet</title></head>");  
        out.println("<body>");  
        out.println("<h1>Autenticação de utilizadores</h1>");  
        out.println("<p>Estou a falar com o utilizador: " + request.getRemoteUser() + "</p>");  
        out.println("<p>Que está no host: " + request.getRemoteHost() + "</p>");  
        out.println("</body></html>");  
    }  
}
```

O Browser é responsável por enviar a informação referente ao utilizador. As definições de segurança normalmente não o permitem.





# Estado entre Pedidos

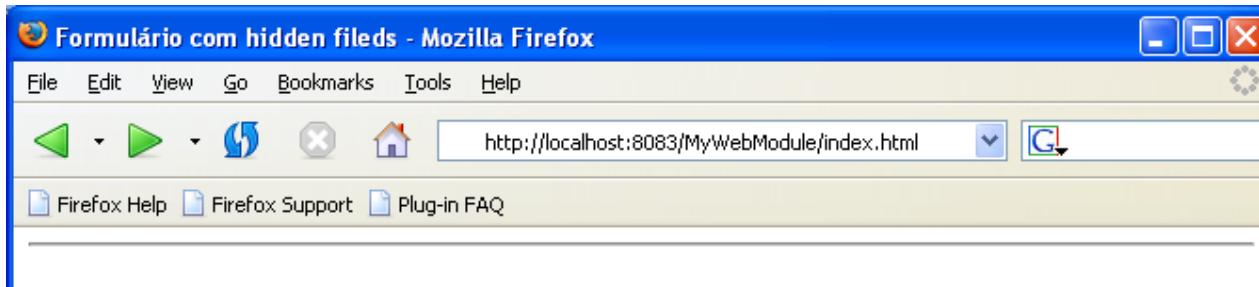
## ■ Campos escondidos:

- Neste caso a informação de estado é mantida como atributos de um formulário mas os atributos estão escondidos;
- Do ponto de vista de quem processa os dados não existe distinção entre processar um campo não escondido e um escondido.





# Estado entre Pedidos



## Formulário com um hidden field a indicar o

```
<html><head><title>Formulário com hidden fields</title></head>
<body><hr/>

<font color="#000080"><h1>Formulário com um hidden field a indicar o estado no fluxo de
p&aacute;ginas</h1></font>

<form method="GET" action="camposescondidosservlet">

<h1>Insira o seu username:</h1><input type="text" name="username"/>

<h1>Insira a sua password:</h1><input type="password" name="password"/>

<input type="hidden" name="estado" value="login"/>

<input type="submit" value="Submeter"/>
<input type="reset" value="Limpar"/>
</form>
</body></html>
```

Hidden field com a informação que o utilizador  
agora está na página de login.



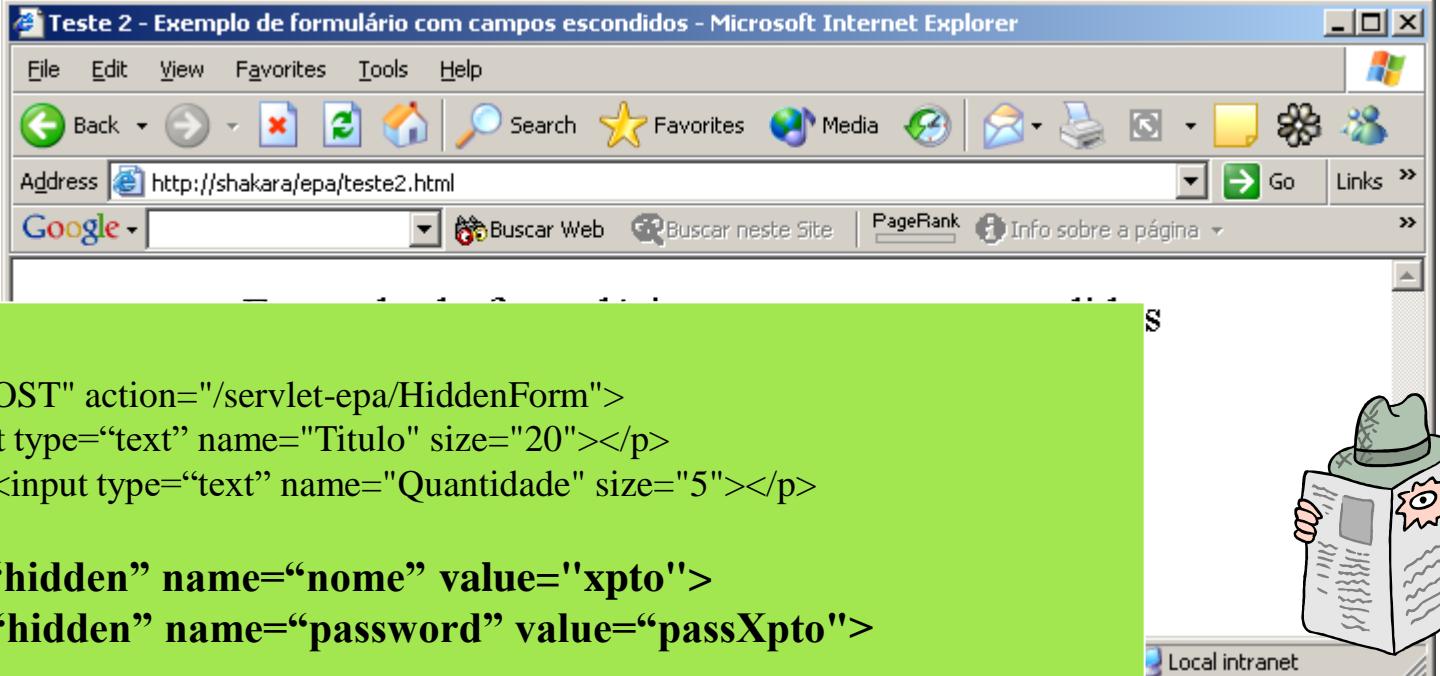
# Estado entre Pedidos

```
public class CamposEscondidosServlet extends HttpServlet {  
    //Process the HTTP Get request  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
            throws ServletException, IOException {  
        String username = request.getParameter("username");  
        if (username == null) { username = ""; }  
        String password = request.getParameter("password");  
        if (password == null) { password = ""; }  
        String estado = request.getParameter("estado");  
        if (estado == null) { estado = ""; }  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println("<html>");  
        out.println("<head><title>CamposEscondidosServlet</title></head>");  
        out.println("<body>");  
        out.println("<h1>Exemplo de campos escondidos</h1>");  
        out.println("<h2>Recebi os seguintes atributos na mensagem http:</h2>");  
        out.println("<p>Username: "+ username +"</p>");  
        out.println("<p>Password: "+ password +"</p>");  
        out.println("<p>Estado: "+ estado +"</p>");  
        out.println("</body></html>");  
    }  
}
```



# Estado entre Pedidos

...  
<form method="POST" action="/servlet-epa/HiddenForm">  
  <p>Titulo: <input type="text" name="Titulo" size="20"></p>  
  <p>Quantidade: <input type="text" name="Quantidade" size="5"></p>  
  
  **< input type="hidden" name="nome" value="xpto">**  
  **< input type="hidden" name="password" value="passXpto">**  
  
  <p><input type="submit" value="Submit">&ampnbsp <input type="reset" value="Reset"></p>  
</form>  
...



Local intranet



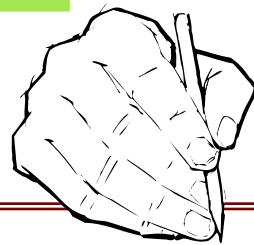
# Estado entre Pedidos

## ■ Rescrita dos URLs:

■ Com esta técnica modifica-se ou rescreve-se o URL acedido:

- © *extra path information;*
- © parâmetros adicionais;
- © URL costumizados;

```
http://shakara/servlet-epa/URLRewritting  
http://shakara/servlet-epa/URLRewritting/476  
http://shakara/servlet-epa/URLRewritting?IdSessao=476  
http://shakara/servlet-epa/URLRewritting;$IdSessao$476
```





# Estado entre Pedidos

## ■ Utilização de *cookies*:

- Um *cookie* não é mais do que um conjunto de dados que é enviado do servidor Web para o *browser* e que mais tarde é reenviado para o servidor pelo *browser*;
- Um *cookie* pode identificar, inequivocamente, um cliente;

## ■ Utilização de *cookies* (restrições):

- Um *browser* pode não aceitar *cookies*;
- Os *browsers* podem limitar-se a aceitar apenas 20 *cookies* por servidor, num total de 300 por utilizador;
- Os *browser* podem limitar a dimensão dos *cookies* a 4096 bytes (4 Kb).





# Exemplo de Utilização de *cookies*

Utilização de *cookies* - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8080/MyWebModule/cookies.html

Firefox Help Firefox Support Plug-in FAQ

Utilização de *cookies*

*Set dos valores*

Nome:

Idade:

*Get dos valores*

```
graph LR; Top[Utilização de cookies - Mozilla Firefox] -- "Yellow lightning bolt" --> Bottom[Utilização de cookies - Mozilla Firefox]
```

Utilização de *cookies* - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8080/MyWebModule/getcookies

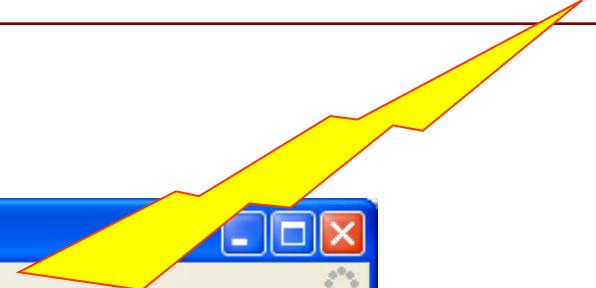
Firefox Help Firefox Support Plug-in FAQ

Não existem cookies





# Exemplo de Utilização de *cookies*



**Utilização de cookies - Mozilla Firefox**

File Edit View Go Bookmarks Tools Help

Back Forward Stop Home http://localhost:8080/MyWebModule/setcookies

Firefox Help Firefox Support Plug-in FAQ

## Dados da sessão:

Nome: Diogo Remédios

Idade: 25

Id: 1ec8909:ff1fbfb0d1b:-7ffe

Id codificado : 1ec8909%3Aff1fbfb0d1b%3A-7ffe

Done





# Exemplo de Utilização de *cookies*

**Utilização de cookies - Mozilla Firefox**

File Edit View Go Bookmarks Tools Help

http://localhost:8080/MyWebModule/cookies.html

Utilização de *cookies*

*Set dos valores*

Nome: Diogo Remédios

Idade: 25

Set Cookie

Done

*Get dos valores*

**Utilização de cookies - Mozilla Firefox**

File Edit View Go Bookmarks Tools Help

http://localhost:8080/MyWebModule/getcookies

Utilização de *cookies*

*Lista de cookies*

Nome do cookie: ID

Valor do cookie: 1ec8909%3Aff1bfb0d1b%3A-7ffe

Nome: Diogo Remédios

Idade: 25

Done

A yellow arrow points from the 'Get dos valores' link in the first window to the 'Lista de cookies' section in the second window, indicating the retrieval process.



# Exemplo de Utilização de *cookies*

```
public class SetCookies extends HttpServlet {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        String nome = request.getParameter("Nome");  
        String idade = request.getParameter("Idade");  
        UserData userData = new UserData(nome, idade);  
        PrintWriter output = response.getWriter();  
        response.setContentType("text/html");  
        Cookie cookieId = new Cookie("ID", userData.getEncodedId());  
        response.addCookie(cookieId);  
  
        output.println("<HTML><HEAD><TITLE>Utilização de cookies</TITLE></HEAD>");  
        output.println("<BODY><H1>Dados da sessão:</H1>");  
        output.println("<P>Nome: " + nome + "</P>");  
        output.println("<P>Idade: " + idade + "</P>");  
        output.println("<P>Id: " + userData.getId() + "</P>");  
        output.println("<P>Id codificado : " + userData.getEncodedId() + "</P>");  
        output.println("</BODY></HTML>");  
    }  
}
```





# Exemplo de Utilização de *cookies*

```
public class GetCookies extends HttpServlet {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        PrintWriter output = response.getWriter();  
        response.setContentType("text/html");  
        output.println("<HTML>");  
        output.println("<HEAD><TITLE>Utilização de cookies</TITLE></HEAD>");  
        output.println("<BODY>");  
  
        Cookie[] cookies = request.getCookies();  
        if (cookies==null)  
            output.println("<H1>Não existem cookies</H1>");  
        else {  
            output.println("<H1>Lista de cookies</H1>");  
            for(int indexCookie=0; indexCookie<cookies.length; ++indexCookie) {  
                output.println("<P>Nome do cookie: " + cookies[indexCookie].getName());  
                output.println("<P>Valor do cookie: " + cookies[indexCookie].getValue());  
                UserData userData = new UserData( cookies[indexCookie].getValue() );  
                output.println("<P>Nome: " + userData.getNome() + "</P>");  
                output.println("<P>Idade: " + userData.getIdade() + "</P>");  
            }  
        }  
        output.println("</BODY>");  
        output.println("</HTML>");  
    }  
}
```





# Estado entre Pedidos - *Session Tracking*

## ■ Usando a API (*Session Tracking*):

- Qualquer servidor que suporte Servlets deve suportar a API “*Session Tracking*”;
  - A implementação mínima para a versão JSDK 2.0 é suportada com base em *cookies*.
- 
- É criada uma instância da classe ***javax.servlet.http.HttpSession*** que funciona como um *container* que fica residente no servidor. Guarda todo o tipo de dados que forem necessários:
    - Informação do perfil do utilizador;
    - Preferências do utilizador;
    - Seleções efetuadas, por exemplo, produtos num carrinho de compras;





# Estado entre Pedidos - *Session Tracking*

## ■ Métodos mais relevantes na *javax.servlet.http.HttpSession*

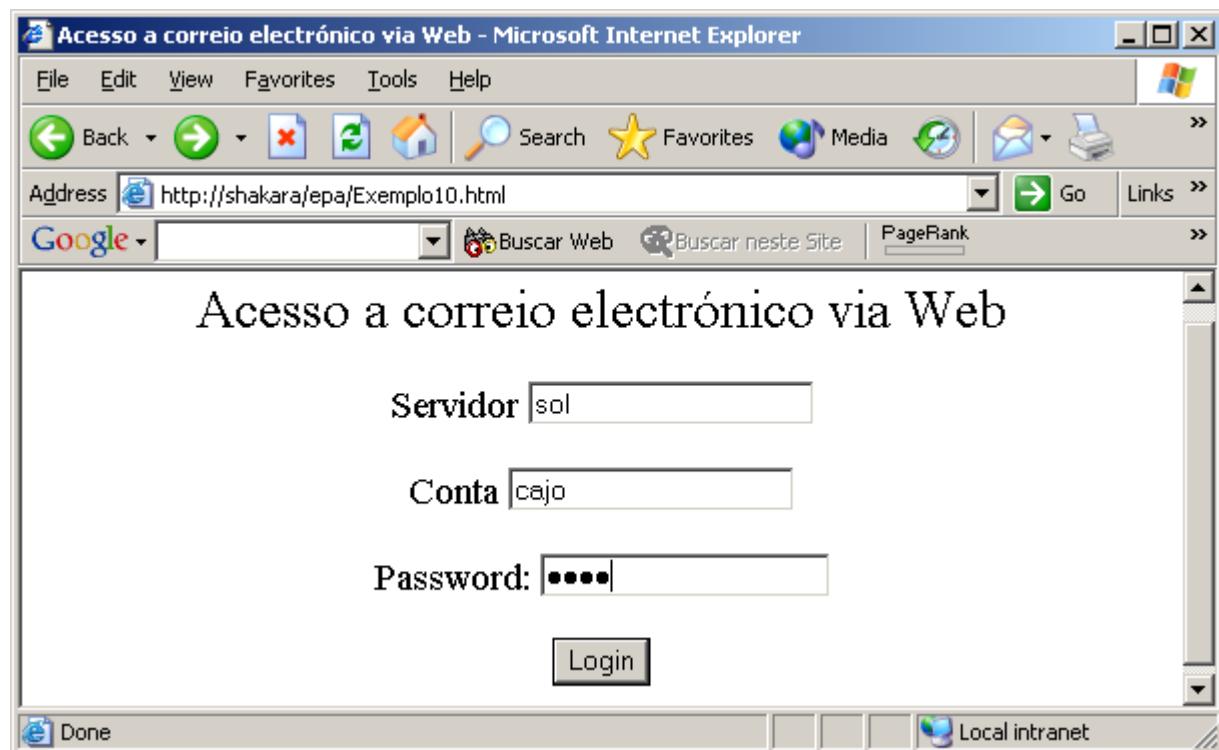
- **java.lang.String getId();** Retorna o identificador único atribuído à sessão.
- **long getCreationTime();** Retorna a data de criação da sessão.
- **java.lang.Object getAttribute(java.lang.String name);** Retorna o atributo guardado com a chave name.
- **java.util.EnumerationgetAttributeNames();** Retorna um enumerado com todos os nomes dos atributos guardados numa dada sessão.
- **void setAttribute(java.lang.String name, java.lang.Object value);**  
Insere na sessão um objecto (value) associado à chave (name).
- **void removeAttribute(java.lang.String name);** Remove o atributo com a chave indicada.
- **boolean isNew();** Indica se a sessão foi criada como resultado.
- **void invalidate();** Invalida a sessão libertando as referências para os objectos que esta contém.





# Exemplo - Acesso a Correio Electrónico

- Pretende-se aceder a uma conta de correio electrónico usando um conjunto de páginas Web.





# Exemplo - Acesso a Correio Electrónico

POST /servlet-epa/WebEmail HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, \*/\*

Referer: http://shakara/epa/Exemplo10.html

Accept-Language: pt

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)

Host: shakara

Content-Length: 36

Connection: Keep-Alive

Cache-Control: no-cache

Servidor=sol&Nome=cajo&Password=cajo



Efectua pedido





# Exemplo - Acesso a Correio Electrónico

HTTP/1.1 200 OK

Server: Netscape-Enterprise/4.1

Date: Tue, 10 Sep 2002 10:35:38 GMT

Set-cookie: NSES40Session=2%253A3d7dcafa%253Aab3fde886430d3f6;path=/;  
expires=Tue, 10-Sep-2002 11:05:38 GMT

Content-type: text/html

Content-length: 829

<HTML>

...

</HTML>



← **Obtém resposta**





# Exemplo - Acesso a Correio Electrónico

The screenshot shows a Microsoft Internet Explorer window titled "Web Email access to server sol - Microsoft Internet Explorer". The address bar contains "http://shakara/servlet-epa/WebEmail". The main content area displays the heading "Gestão da conta *cajo*". Below it is a form titled "Opções:" with three radio button options: "Visualizar mensagens" (selected), "Criar Mensagem", and "Terminar". A "Send" button is located at the bottom left of the form. A large yellow lightning bolt-shaped arrow points from the "Send" button towards the bottom right corner of the slide.

Gestão da conta *cajo*

Opções:

Visualizar mensagens  
 Criar Mensagem  
 Terminar

Send

Done Local intranet



# Exemplo - Acesso a Correio Electrónico

POST /servlet-epa/WebEmail HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, \*/\*

Referer: http://shakara/servlet-epa/WebEmail

Accept-Language: pt

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)

Host: shakara

Content-Length: 10

Connection: Keep-Alive

Cache-Control: no-cache

Cookie: NSES40Session=2%253A3d7dcafa%253Aab3fde886430d3f6

Opcoes=Vis



Efectua pedido





# Exemplo - Acesso a Correio Electrónico

HTTP/1.1 200 OK

Server: Netscape-Enterprise/4.1

Date: Tue, 10 Sep 2002 10:47:01 GMT

Set-cookie: NSES40Session=2%253A3d7dcafa%253Aab3fde886430d3f6;path=/;  
expires=Tue, 10-Sep-2002 11:17:02 GMT

Content-type: text/html

Content-length: 149

```
<HTML>
<HEAD><TITLE>Web Email access</TITLE></HEAD>
<BODY>
<H1>User Name: cajo</H1>
<H1>Password: cajo</H1>
</BODY>
</HTML>
```



Obtém resposta





# Exemplo - Acesso a Correio Electrónico





# Exemplo - Acesso a Correio Electrónico

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Acesso a correio electrónico através de um browser.
 */
public class WebEmail extends HttpServlet {
    PrintWriter output;
    HtmlGenerator page;
    HttpSession session;
```



# Exemplo - Acesso a Correio Electrónico

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    output = response.getWriter();
    page   = new HtmlGenerator( output );
    session = request.getSession(true);

    response.setContentType("text/html");
    Boolean validUser = (Boolean)session.getAttribute("WebEmail.validUser");
    if ( (validUser==null) || (validUser.booleanValue()==false) )
        CheckLogin(request, response);
    else
        ProcessEmail(request, response);
}
```



# Exemplo - Acesso a Correio Electrónico

```
private void CheckLogin(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String user      = request.getParameter("Nome");
    String password = request.getParameter("Password");
    String server   = request.getParameter("Servidor");

    Pop3Client popClient = new Pop3Client(server, user, password);
    if (popClient.isUserOk()) {
        session.setAttribute("WebEmail.validUser", new Boolean(true));
        session.setAttribute("WebEmail.user", user);
        session.setAttribute("WebEmail.password", password);
    }
}
```



# Exemplo - Acesso a Correio Electrónico

```
 . . .
page.beginPage("Web Email access to server " + server);
page.h1("Gestão da conta <i>" + user +"</i>");

page.beginForm("POST", "/servlet-epa/WebEmail");
buildForm(); page.endForm(); page.endPage();

}

else {
    session.setAttribute("WebEmail.validUser", new Boolean(false) );
    response.setHeader("Refresh", "3; URL=/epa/Exemplo10.html");
    page.beginPage("Web Access to server Krillin");
    page.h1("Bad user name or password."); page.endPage();
}
}
```



# Exemplo - Acesso a Correio Electrónico

```
private void ProcessEmail(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String user      = (String) session.getAttribute("WebEmail.user");
    String password = (String) session.getAttribute("WebEmail.password");

    page.beginPage("Web Email access");

    page.h1("User Name: " + user);
    page.h1("Password: " + password);

    page.endPage();
}

};
```



# Exemplo - Loja online - WebShop

WebShop - Compras online - Mozilla Firefox  
File Edit View Go Bookmarks Tools Help  
http://localhost:8080/MyWebModule/webshop

## WebShop - Compras online

Criei o Shopping cart na sessao: AAE479B1F3774AA92E7E1BDDE

[Ver Carrinho de compras](#)

### Lista de Produtos:

Produto:	Preço:
LCDTV SAMSUNG LW15M13C	57.00 EUR
Placa de Rede Zaapa Wireless 11Mbps USB	39.99 EUR
Portátil Acer TM291LMi	1419.00 EUR
Leitor DVD Philips DVD 737 DIVX	179.90 EUR
Plasma NEC PX-42VP4	4099.90 EUR
DVD+RW/-RW Plextor 708A	289.00 EUR
HP DVD300e Externo 2,4x/4x/16x/40x	299.00 EUR
Placa de Rede US Robotics 100Mbps Wireless PCI	99.99 EUR
Camara Fotografica Digital Canon	459.70 EUR

Done

WebShop - Compras online - Mozilla Firefox  
File Edit View Go Bookmarks Tools Help  
http://localhost:8080/MyWebModule/webshop?estado=ListaCarrinho

## WebShop - Compras online

[Voltar ao Menu](#)

### ■ Carrinho de Compras ■ :

*O seu carrinho ainda está vazio*

[Voltar ao Menu](#)

Secção de Engenharia de Sistemas  
Autoria: Eng. Diogo Remédios  
Done

WebShop - Compras online - Mozilla Firefox  
File Edit View Go Bookmarks Tools Help  
http://localhost:8080/MyWebModule/webshop?estado=Adicionar&prc

## WebShop - Compras online

### O produto:

*Placa de Rede US Robotics 100Mbps Wireless PCI [99.99 EUR]*

Foi adicionado ao seu carrinho de compras.

[Voltar ao Menu](#)

Done



# Exemplo - WebShop

WebShop - Compras online - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8080/MyWebModule/webshop?estado=Menu

Firefox Help Firefox Support Plug-in FAQ

## WebShop - Compras online

Ver Carrinho de compras

### Lista de Produtos:

Produto:	Preço:
LCDTV SAMSUNG LW15M13C	57.00 EUR
Placa de Rede Zaapa Wireless 11Mbps USB	39.99 EUR
Portátil Acer TM291LMI	1419.00 EUR
Leitor DVD Philips DVD 737 DIVX	179.90 EUR
Plasma NEC PX-42VP4	4099.90 EUR
DVD+RW/-RW Plextor 708A	289.00 EUR
HP DVD300e Externo 2,4x/4x/16x/40x	299.00 EUR
Placa de Rede US Robotics 100Mbps Wireless PCI	99.99 EUR
Camara Fotografica Digital Canon PowerShot A80	458.70 EUR

Adicionar ao carrinho de compras  
X - Remover do carrinho de compras

Done

WebShop - Compras online - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8080/MyWebModule/webshop?estado=ListaCarrinho

Firefox Help Firefox Support Plug-in FAQ

## WebShop - Compras online

Voltar ao Menu

### Carrinho de Compras :

Produto:	Preço:
Placa de Rede US Robotics 100Mbps Wireless PCI	99.99 EUR

Total = 99.99 EUR

X - Remover do carrinho de compras

Voltar ao Menu

Done



# Exemplo - WebShop

- Pretende-se aceder a uma loja online usando um conjunto de páginas Web.
- Neste exemplo é guardado na sessão um objecto que é o carrinho de compras. Este contém um vector com os vários itens escolhidos.

■ A leitura dos produtos existentes é feita no init().

```
...  
  
public void init(){ // ler do ficheiro WebShop.txt e preencher o itemVector  
    try {  
        persistentDir = new String(getServletContext().getRealPath("/") );  
        BufferedReader bufferedReader = new BufferedReader(new FileReader( persistentDir + "WebShop.txt"));  
        String s;  
        while((s = bufferedReader.readLine()) != null){  
            //ler a contagem e o nome da musica  
            String[] res = s.split("_sep_");  
            Item product = new Item(res[0], res[1], res[2]);  
            itemVector.add(product); //adicionar o novo produto  
        }  
        bufferedReader.close();  
    }  
    catch (Exception e){ log("Excecao a ler a os produtos do ficheiro WebShop.txt..."); }  
}
```

itemVector é global para que todas as threads que executem o service possam saber quais os produtos existentes na loja.



# Exemplo - WebShop

- Caso seja o primeiro acesso criar o carrinho de compras e guardar na sessão.

```
...  
  
public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
    HttpSession mySession = request.getSession(true);  
    PrintWriter output = response.getWriter();  
    response.setContentType("text/html");  
  
    ...  
  
    if(mySession.getAttribute("WebShop.ShoopingKart") == null){  
        // there is no shooing kart, let's create one  
        ShoopingKart sk = new ShoopingKart();  
        mySession.setAttribute("WebShop.ShoopingKart", sk);  
        output.println("<p> Criei o Shooing kart na sessao: "+ mySession.getId() +"</p>");  
    }  
  
    ...
```

**Primeiro acesso. O carrinho ainda não existe na sessão.**

**O carrinho de compras é inserido na sessão. A partir deste momento o cliente deverá ter sempre este carrinho com as suas compras dentro.**



# Exemplo - WebShop

- Sempre que é necessário o acesso ao carrinho das compras é utilizada a sessão.

```
... // exemplo de adicionar um produto ao carrinho

if ((request.getParameter("estado").compareTo("Adicionar") == 0)
    && (request.getParameter("productId") != null) ) {

    // adicionar produto ao carrinho

    String prodId = request.getParameter("productId");

    ShoopingKart mySk = (ShoopingKart) mySession.getAttribute("WebShop.ShooingKart");
    Iterator existingItemIter = itemVector.iterator();

    while(existingItemIter.hasNext()){

        Item existingItem = (Item) existingItemIter.next();
        if(existingItem.getId().compareTo(prodId)==0){           //produto a adicionar

            mySk.add(existingItem);
            output.println("<h1>O produto:</h1>");
            output.println("<h2>&ampnbsp&ampnbsp <i><font size=\"6\">&ampnbsp<font color=\"#008000\\\">" +
                           + existingItem.getName() + "</font></font></i> <span style=\"font-weight: " +
                           +"400; font-style: italic\\\"><font size=\"4\\\">&ampnbsp&ampnbsp[" +
                           + existingItem.getPrice() +" EUR]</font></span></h2>" +
                           +"<h2>Foi adicionado ao seu carrinho de compras.</h2>" +
                           +"<p><a href=\"webshop?estado=Menu\\\">Voltar ao Menu</a></p>");

            break;
        }
    } // Element Added to the shooping kart
} ...
```

Obter carrinho de compras

Adição de um produto ao carrinho



# Conclusões

- As Servlets são uma alternativa no desenvolvimento de páginas dinâmicas geradas no contexto de um servidor Web;
- Existem outras alternativas:
  - Programas CGI;
  - Extensões à API do servidor;
  - Active Server Pages – ASP;
  - JavaServer Pages – JSP;
  - JavaScript suportado pelo servidor.
- Não necessitam que o *browser* disponha de uma máquina virtual Java;
- São portáveis uma vez que estão escritas em Java e obedecem a uma API conhecida;



# Conclusões

- Podem usar todo o potencial da linguagem Java;
- São eficientes. Uma vez carregada em memória a Servlet fica disponível para satisfazer mais pedidos;
- São seguras:
  - Strong type check da linguagem Java;
  - Utilização de exceções;
  - Utilização de Security Managers.
- O código desenvolvido é simples. A própria API disponibiliza objectos específicos para certas funcionalidades;



# Conclusões

- As Servlets podem ser vistas como uma extensão ao servidor;
- Como estão integradas no servidor permitem uma maior interacção entre a Servlet e o servidor Web;
- O código é interpretado, o que, face a outras alternativas pode fazer com que a Servlet tenha um desempenho pior caso a implementação não recorra à tecnologia JIT;
- A utilização de Servlets pode não ser uma boa opção na geração de formulários se a quantidade de informação gerada for muito grande.



# Lista de Exemplos Demonstrativos

- Índice [Exemplos de servlets disponíveis na PHOENIX, na sua área de web]
- *Hello World:* [HelloWorldServlet\  
]
  - © Simples; [HelloWorldServlet.java]
  - © Com formulário para recolher o nome; [HelloToServlet.java]
- *Concorrência num contador de acessos:* [ConcorrenciaServlet\  
]
  - © Servlet Problemática; [ServletErrada.java]
  - © Solução 1; [ServletBoa1.java]
  - © Solução 2; [ServletBoa2.java]
- *Listar cabeçalhos http:* [SnoopingHttpHeadersServlet\  
]
  - © Lista todos os cabeçalhos http; [SnoopingHttpHeadersServlet.java]
  - © Lista alguns cabeçalhos http; [SnoopingHeaderCompCGIServlet.java]



# Lista de Exemplos Demonstrativos

- *Manutenção de informação de estado:* [ManutencaoDeEstado\  
]
  - ⌚ Autenticação de Utilizadores; [AutenticacaoDeUtilizadoresServlet.java]
  - ⌚ Campos escondidos; [CamposEscondidosServlet.java]
  - ⌚ Rescrita do URL; [index.html]
  - ⌚ Cookies; [setcookies.java e getcookies.java]
  
- *WebShop :* [WebShopServlet\  
]
  - ⌚ Exemplo guardando informação na sessão; [WebShop.java]  
[ShoopingKart.java]  
[Item.java]



# Referências de Estudo

- Servlet API e J2SE 1.4.2 API
  - <http://www.deetc.isel.ipl.pt/engenhariadesi/cadeiras/scd/documentacao.htm>
- Cookies Netscape
  - [http://www.netscape.com/newsref/std/cookie\\_spec.html](http://www.netscape.com/newsref/std/cookie_spec.html)
- API's de Servidores WEB
  - <http://www.apache.org/docs/API.html>
- Classes utilitárias
  - <http://www.acme.com/java/software/>



# Glossário de Termos

- WWW – World-Wide Web;
- HTML – HyperText Markup Language;
- HTTP – HyperText Transfer Protocol;
- URL – Uniform Resource Locator;
- URI – Uniform Resource Identifier;
- URN – Uniform Resource Name;
- RFC – Request For Comments;
- CGI – Common Gateway Interface;
- API – Application Programming Interface;
- ASP – Active Server Pages;
- JSP – JavaServer Pages;



# Bibliografia

- RFC's:
  - 1945 – HTTP/1.0;
  - 2068 – HTTP/1.1;
  - 2109 – *HTTP State Management Mechanism*;
- HTML Sourcebook, 3<sup>th</sup> Edition;
  - Graham, Ian S. – Wiley Computer Publishing
- Java Servlet Programming;
  - Hunter, J. & Crawford, W – O' Reilly
- Professional JSP 2nd Edition
  - Karl Avedal, Robert Burdick, ..., Steve Wilkinson
  - Publisher: Wrox Press Ltd, ISBN: 1861004958

